

# Classical Computational Geometry in GeomNet\*

Gill Barequet<sup>†</sup>   Stina S. Bridgeman<sup>‡</sup>   Christian A. Duncan<sup>†</sup>   Michael T. Goodrich<sup>†</sup>  
Roberto Tamassia<sup>†</sup>

## Abstract

In this paper we present *GeomNet*, a system for performing distributed geometric computing over the Internet. We also provide several examples of actual geometric algorithms that our system already supports. Application domains for GeomNet include collaborative research and distance education.

**Keywords:** geometric computing, Internet computing, cooperative computing, visualization.

## 1 Introduction

Researchers in computational geometry are developing a host of efficient algorithms for solving geometric problems, and many are taking the design of these algorithms all the way to working implementations. This implementation component is no small task, for computational geometry programs must deal with many issues that are not faced in other software applications. This is largely due to the fact that geometric data typically involves a non-trivial combination of numerical and combinatorial relationships, and much care must be taken to develop formats for representing geometric data and for performing robust and reliable computations using such representations. Fortunately, there is a rich, growing literature directed at developing techniques for dealing with such robustness issues [1, 7, 9, 10, 11, 12, 13, 15, 18, 21]. Moreover, fundamental geometric algorithms are being redesigned taking into account robustness and arithmetic precision issues [9, 16].

\*The work of all five authors is supported in part by the U.S. Army Research Office under grant DAAH04-96-1-0013. In addition, work of the second author is supported in part by a National Science Foundation Graduate Fellowship, work of the fourth author is supported in part by the National Science Foundation under grant CCR-9625289, and work of the fifth author is supported in part by the National Science Foundation under grant CCR-9423847.

<sup>†</sup>Center for Geometric Computing, Dept. of Computer Science, Johns Hopkins University, Baltimore, MD 21218. E-mail: [barequet|duncan|goodrich]@cs.jhu.edu

<sup>‡</sup>Center for Geometric Computing, Dept. of Computer Science, Brown University, Providence, RI 02912. E-mail: [ssb|rt]@cs.brown.edu

Another important aspect of software development in computational geometry is the visualization of geometric objects and the animation of geometric algorithms, with the dual purpose of demonstration and debugging [14].

## 1.1 Collections of Geometric Software

Collections of geometric software can be found on the Internet at Amenta's "Directory of Computational Geometry Software" site,<sup>1</sup> and a host of applications of such software can be found at Eppstein's "Geometry in Action" site.<sup>2</sup> Indeed, Erickson has an even more extensive collection of geometric software at his "Computational Geometry Software" site, which is actually just a small part of his extensive directory of "Computational Geometry Pages."<sup>3</sup> In addition, there is a large collaborative effort underway to develop an extensive Computational Geometry Algorithms Library (CGAL)<sup>4</sup> [8, 20], which itself is built upon the successful Library of Efficient Data Types and Algorithms (LEDA)<sup>5</sup> [6, 17]. This effort is directed at building a large collection of C++ routines for solving computational geometry problems. Therefore, in spite of the difficulties presented by geometric algorithm implementation, there is now a rich collection of robust implementations of geometric algorithms, and this collection should grow significantly in the years ahead.

Unfortunately, each package of geometric software typically creates its own unique, incompatible data format. Unlike many other domains such as numerical analysis, which have simple, standard notions for describing problems (e.g., using matrices), there are no standard mechanisms for fully representing geometric data. The computational geometry research community could, of course, spend a considerable amount of effort on designing a standard language for describing geometric data, but this does not seem likely. Such an effort would necessitate the encoding of all possible relationships between the numerical and combinatorial components of geometric data, which would be a Herculean task.

We feel that practitioners and researchers can take better advantage of the latest developments in computational geometry if implementations of geometric algorithms are made widely available on the Internet and are fully usable without the learning of new file for-

<sup>1</sup><http://www.geom.umn.edu/software/cglist/>

<sup>2</sup><http://www.ics.uci.edu/~eppstein/geom.html>

<sup>3</sup><http://www.cs.duke.edu/~jeffe/compgeom/>

<sup>4</sup><http://www.cs.ruu.nl/CGAL/>

<sup>5</sup><http://www.mpi-sb.mpg.de/LEDA/leda.html>

mats, calling sequences, or class hierarchies in software libraries. In this paper we present a system designed to address these issues, give examples of applications supported by the current prototype implementation, and discuss further extensions.

## 2 The GeomNet Vision

We envision an “Internet computing” framework, which we call *GeomNet*, where a family of *geometric computing servers* execute a variety of geometric algorithms on behalf of remote clients, which can be either users interacting through a Web browser interface, or application programs connecting directly through sockets. Such an environment could be used in a variety of scenarios, including execution of algorithms on behalf of remote clients, checking geometric structures for consistency, experimental study and comparison of algorithms, design of new algorithms through plug-and-play integration of existing algorithms, demonstrations in an educational setting, interactive electronic books and manuals, and electronic commerce for CAD and GIS companies.

### 2.1 Cooperative Computing

The goal for GeomNet is a networked geometric computing environment whose functionality is independent of whether the user is interfacing with it through the Internet on a notebook computer or locally on a high-end graphics workstation. Thus, our system is based on a layered client-server architecture. The system (with minimal user guidance) forwards difficult computations to a compute server, converts between various file formats, performs a requested computation, and sends the results back to the user. We refer to such negotiations between the client and server as *cooperative computing*, with the following scenarios serving as typical levels of computational commitment on the part of the client:

**Level 0.** The client has minimal computational and graphical resources. In this case, all computation, including rendering, must be performed on the server and transferred to the client as images or display primitives.

**Level 1.** The client has some computational and graphical resources but limited network bandwidth. In this case, non-CPU computations, such as those pertinent to the graphical user interface, should be performed on the client, while CPU-intensive computations should be performed by the server or another powerful computer on the network.

**Level 2.** The client has workstation-quality computational and graphical resources and high network bandwidth. In this case all but the most CPU-intensive computations should be performed on the client, with very CPU-intensive computations still performed by the server or another powerful computer on the network.

**Level 3.** The client has significant computational and graphical resources (but not necessarily very high network bandwidth). In this case the client is better off performing the entire computation locally (with the server performing only code-shipping services).

Levels 0 and 3 are well-developed paradigms in dis-

tributed computing, with Level 0 cooperative computing fitting the X network-transparent window system, and Level 3 cooperative computing fitting the standard Java computing paradigm. Our intention, however, is to define a system that can negotiate with a client to determine the most efficient level of cooperation. The intermediate levels also allow the integration of existing implementations not written in Java as well as enabling users to access code that the authors may not wish to make available, even in compiled form. Thus, our design of GeomNet is concerned mostly with Level 1 and Level 2 cooperative computing.

### 2.2 System Goals

Specific goals of the GeomNet system embodying our vision of a cooperative computing environment include:

1 *Security*: both client and server should be safe from intruder attacks;

2 *Code protection*: code should be protected from software piracy as much as possible;

3 *Authoring*: in order to promote the development of a user community, both new and legacy implementations should be easily integrated into the system;

4 *Efficiency*: computations should be performed quickly and require minimum communication costs;

5 *Correctness*: the system should provide “certificates” of the correctness of its computations; and

6 *Cooperation*: users with all levels of computing power should be able to easily perform interesting geometric computations and visualizations.

Our GeomNet system currently satisfies each of the first four goals to some degree. It provides security inherited from the Java virtual environment, code protection between client and server and between separate components of the server, authoring through a modular design allowing even non-Java packages to be integrated with a simple application “wrapper” mechanism, and efficiency through the integration of well-designed implementations of fast algorithms. With further integration of applications we hope that it will soon also satisfy the latter two design goals as well. Still, even in its current form, the GeomNet environment provides a powerful tool for those applications that interact with geometric data or can benefit from geometric visualizations of non-geometric data.

In addition, we feel that one of the prime strengths of GeomNet is that it complements rather than replaces the traditional model of supporting the development of geometric computing applications by providing software libraries. Indeed, for GeomNet to reach its full potential we need to be able to integrate as many tools for performing geometric computations as is possible (and allowed). A prototype implementation of GeomNet is available on the Internet through the support of the Center for Geometric Computing,<sup>6</sup> with prototype (pre-alpha release) servers available at Brown’s GeomNet site<sup>7</sup> and the GeomNet site<sup>8</sup> at Johns Hopkins. New releases are planned throughout 1997.

<sup>6</sup><http://www.cs.brown.edu/cgc/>

<sup>7</sup><http://loki.cs.brown.edu:8081/geomNet/>

<sup>8</sup><http://www.cgc.cs.jhu.edu/geomNet/>

### 3 Applications

Having covered some of the features of GeomNet, let us now discuss some example applications.

**Fundamental Algorithms.** One of the prime applications that we envision is for GeomNet to be used to invoke implementations of fundamental geometric algorithms. We have therefore included in our prototype implementations of several such fundamental algorithms, including 2-D and  $d$ -D convex hulls, Delaunay triangulations and Voronoi diagrams, and width of a  $d$ -D point set. We have developed several interface mechanisms for invoking these routines. The simplest interface is the interactive applet interface designed to gain intuition about geometric algorithms as well as for testing GeomNet servers. For handling large geometric data files we provide an HTML forms interface, and for production-level usage we provide a socket interface which may be invoked by an application program running on the client. The output from the server can be immediately sent to an interactive display system running on the client or piped into a client application.

**Graph Drawing Server.** The Graph Drawing tool can be used for drawing graphs for documents or user applications, studying and comparing graph drawing algorithms, translating between formats for describing graphs and their drawings, creating a database of graphs occurring in user applications, and providing demonstrations in an educational setting. The current prototype supports three algorithms for orthogonal drawings of general graphs, and two specialized algorithms for acyclic digraphs and series-parallel digraphs. The user is free to specify any input/output format independently of the algorithm requested.

**Geometric Algorithm Animation.** We are incorporating geometric algorithm animation capabilities into GeomNet (at cooperative computing levels 1 and 2) by extending our previous work on the Mocha model for Web-based algorithm animation [2, 3, 4, 5]. Within the GeomNet client-server architecture, Mocha optimally partitions the software components of a typical algorithm animation system. A variety of animations of fundamental 2-D geometric algorithms implemented within LEDA are available through Mocha.

### 4 Conclusion

We have presented the GeomNet architecture for performing geometric computing over the Internet. We anticipate the availability of a significant variety of applications based on the GeomNet architecture, mainly due to the small amount of effort required for integrating new applications into the modular system. It is our feeling that such a system provides the accessibility of geometric algorithms while overcoming the usual problems of lack of knowledge of where to look for such algorithms, differences in interfaces and in file formats, and unavailability of sufficient computer resources. All these problems are handled by GeomNet, which we consider as a significant step towards sharing geometric data and computing resources.

### References

- [1] F. Aynaim, J.-D. Boissonnat, O. Devillers, F. Preparata, and M. Yvinec. Evaluation of a new method to compute signs of determinants. In *Proc. 11th Ann. ACM Symp. Comp. Geom.*, C16-C17, 1995.
- [2] J. E. Baker, I. F. Cruz, G. Liotta, and R. Tamassia. A new model for algorithm animation over the WWW. *ACM Comp. Surv.*, 27(4):568-572, 1995.
- [3] J. E. Baker, I. F. Cruz, G. Liotta, and R. Tamassia. Algorithm animation over the World Wide Web. In *Proc. Int. Workshop on Advanced Visual Interfaces (AVI '96)*, 203-212, 1996.
- [4] J. E. Baker, I. F. Cruz, G. Liotta, and R. Tamassia. Animating geometric algorithms over the Web. In *Proc. 12th Ann. ACM Symp. Comp. Geom.*, C3-C4, 1996.
- [5] J. E. Baker, I. F. Cruz, G. Liotta, and R. Tamassia. The Mocha algorithm animation system. In *Proc. Int. Workshop on Advanced Visual Interfaces (AVI '96)*, 248-250, 1996.
- [6] C. Burnikel, J. Könnemann, K. Mehlhorn, S. Näher, S. Schirra, and C. Uhrig. Exact geometric computation in LEDA. In *Proc. 11th Ann. ACM Symp. Comp. Geom.*, C18-C19, 1995.
- [7] H. Edelsbrunner and E. P. Mücke. Simulation of simplicity: a technique to cope with degenerate cases in geometric algorithms. *ACM Trans. Graph.*, 9:66-104, 1990.
- [8] A. Fabri, G.-J. Giezeman, L. Kettner, S. Schirra, and S. Schoenherr. The CGAL kernel: A basis for geometric computation. In *Applied Computational Geometry (Proc. WACG '96)*, Lecture Notes in Computer Science. Springer-Verlag, 1997.
- [9] S. Fortune. Numerical stability of algorithms for 2-d Delaunay triangulations. *Int. J. Comp. Geom. Appl.*, 5(1):193-213, 1995.
- [10] S. Fortune and V. Milenkovic. Numerical stability of algorithms for line arrangements. In *Proc. 7th Ann. ACM Symp. Comp. Geom.*, 334-341, 1991.
- [11] D. H. Greene and F. F. Yao. Finite-resolution computational geometry. In *Proc. 27th Ann. IEEE Symp. Found. Comp. Sci.*, 143-152, 1986.
- [12] L. Guibas and D. Marimont. Rounding arrangements dynamically. In *Proc. 11th Ann. ACM Symp. Comp. Geom.*, 190-199, 1995.
- [13] L. J. Guibas, D. Salesin, and J. Stolfi. Epsilon geometry: building robust algorithms from imprecise computations. In *Proc. 5th Ann. ACM Symp. Comp. Geom.*, 208-217, 1989.
- [14] A. Hausner and D. Dobkin. Making geometry visible: An introduction to the animation of geometric algorithms. In J.-R. Sack and J. Urrutia, editors, *Handbook on Computational Geometry*, North Holland, 1997. To appear.
- [15] C. M. Hoffmann, J. E. Hopcroft, and M. S. Karasick. Towards implementing robust geometric computations. In *Proc. 4th Ann. ACM Symp. Comp. Geom.*, 106-117, 1988.
- [16] G. Liotta, F. P. Preparata, and R. Tamassia. Robust proximity queries in implicit Voronoi diagrams. TR CS-96-16, Center for Geometric Computing, CS Dept., Brown Univ., Providence, RI, 1996.
- [17] K. Mehlhorn and S. Näher. LEDA: a platform for combinatorial and geometric computing. *Commun. ACM*, 38:96-102, 1995.
- [18] V. Milenkovic. Double precision geometry: a general technique for calculating line and segment intersections using rounded arithmetic. In *Proc. 30th Ann. IEEE Symp. Found. Comp. Sci.*, 500-505, 1989.
- [19] T. Munzner, S. Levy, and M. Philips. Geomview: A system for geometric visualization. In *Proc. 11th Ann. ACM Symp. Comp. Geom.*, C12-C13, 1995.
- [20] M. H. Overmars. Designing the computational geometry algorithms library CGAL. In *Applied Computational Geometry (Proc. WACG '96)*, Lecture Notes in Computer Science. Springer-Verlag, 1997.
- [21] C. K. Yap. A geometric consistency theorem for a symbolic perturbation scheme. *J. Comp. Syst. Sci.*, 40:2-18, 1990.